Feature Construction for Time Ordered Data Sequences

Michael Schaidnagel
School of Computing
University of the West of Scotland
Email: B00260359@studentmail.uws.ac.uk

Fritz Laux
Faculty of Computer Science
Reutlingen University
Email: fritz.laux@reutlingen-university.de

***Abstract*—The recent years and especially the Internet have changed the way on how data is stored. We now often store data together with its creation time-stamp. These data sequences potentially enable us to track the change of data over time. This is quite interesting, especially in the e-commerce area, in which classification of a sequence of customer actions, is still a challenging task for data miners. However, before standard algorithms such as Decision Trees, Neuronal Nets, Naive Bayes or Bayesian Belief Networks can be applied on sequential data, preparations need to be done in order to capture the information stored within the sequences. Therefore, this work presents a systematic approach on how to reveal sequence patterns among data and how to construct powerful features out of the primitive sequence attributes. This is achieved by sequence aggregation and the incorporation of time dimension into the feature construction step. The proposed algorithm is described in detail and applied on a real life data set, which demonstrates the ability of the proposed algorithm to boost the classification performance of well known data mining algorithms for classification tasks.**

***Index Terms*—feature construction, sequential data, temporal data mining**

I. Introduction

Huge amounts of data are being generated on a daily basis, in almost every aspect of our live. Advancements in computer science as well as computer hardware enable us to store and analyze these data. Especially in the e-commerce area it is common to log all user activities in an online shop. Such data can be ordered by their timestamp and can be allocated to data sequences of particular users. However, the logged activities or actions are not stored in a form that enables data mining right away. Therefore, it is important to pre-process the data before analyzing it (see also Han [1], Liu [2]). When data is only represented by primitive attributes and there is no prior domain expert knowledge available, the pre-processing task becomes challenging and creates the need for automated techniques. At this point attribute selection and/or feature construction techniques need to be applied. Attribute selection can be defined as the task of selecting a subset of attributes, which are able to perform at least as good on a given data mining task as the primitive (original) attributes set. The original values of the data set are called attributes, while the constructed data are called features. It is possible that primitive attributes are not able to adequately describe eventually existing relations among primitive attributes. Such interrelations (or also called interactions, see Shafti [3]) can occur in a data set, if the relation between one attribute and the target concept depends on another attribute (see also Shafti [4]). Attribute selection alone can fail to find existing interaction among data. Therefore, one goal for feature construction is to find and highlight interactions. Feature construction can be defined as the process of creating new compound properties using functional expressions on the primitive attributes. Shafti [3] distinguishes between two types of features construction techniques in terms of their construction strategy:

- hypothesis-driven: create features based on a hypothesis (which is expressed as a set of rules). These features are then added to the original data set and are used for the next iteration in which a new hypothesis will be tested. This process continues until a stopping requirement is satisfied.
- data-driven methods: create features based on pre-determined functional expressions, which are applied on combinations of primitive features of a data set. These strategies are normally non-iterative and the new features are evaluated by directly assessing the data.

A. Problem description

Both feature construction strategies are not able to include a dimension that is unique to sequential data: the time elapsed between the corresponding actions. The so far described strategies are not able to express a pattern, which occurs in the course of time. Reason for this is their focus on tuples (rows) in a database. In this work, we will focus on the data-driven strategy and propose a new technique that is able to find patterns that are spread across several rows of a sequence. This can be achieved by creating meaningful features that are able to transform sequence information into the tuple-space.

B. Structure of the paper

The remainder of this work is structured as follows: Section II will give a short overview about the related literature. Subsection II-A will highlight our contribution to the particular research field. Our proposed algorithm takes additional consideration on sequential data. The characteristics of such data is described in Section III. Our approach to feature construction will be described in detail in Section IV. We divided the algorithm into four logical parts, which are respectively described in the subsections of Section IV. This is followed by an

 ISBN: 978-1-61208-334-6

experimental analysis in Section V, in which we demonstrate the ability of our proposed algorithm to boost classification performance on a real life data set. The paper is concluded by the sections Conclusion (Section VI) and Future Work (Section VII).

II. Related Work

Earlier work in the field of feature construction was done by Setiono and Liu [5]. They used a neuronal network to construct features in an automatic way for continuous and discrete data. Pagallo [6] proposed FRINGE, which builds a decision tree based on the primitive attributes to find suitable boolean combinations of attributes near the fringe of the tree. The newly constructed features are then added to the initial attributes set and the process is repeated until no new features are created. Zupan and Bohanec [7] used a neuronal net for attribute selection and applied the resulting feature set on the well known C4.5 [8] induction algorithm. Feature construction can also be used in conjunction with linguistic fuzzy rule models. García [9] et al. use previously defined functions over the input variables in order to test if the resulting combination returns more information about the classification than the single variables.

However, in order to deal with increasing complexity of their genetic algorithm in the empirical part, García only used three functions (SUM{Xi, Xj}, PRODUCT{Xi, Xj}, SUBSTRACT_ABS{Xi, Xj}) to enlarge the feature space. Another approach to feature construction, which utilizes a genetic algorithm, is described by Alfred [11]. Although, his approach is not using different functions to create new combinations of features, it can create a big variety of features since it is not limited to binary combination. That means that it is able to combine more than two attributes at a time. The genetic algorithm selects thereby the crossover points for the feature sequences. Another mentionable contribution to the field of feature construction was done by Shafti [4]. She describes MFE3/GA, a method that uses a global search strategy (i.e., finding the optimal solution) to reduce the original data dimensions and find new non-algebraic representations of features. Her primary focus is to find interactions between the original attributes (such as the interaction of several cards in a poker game that form a certain hand). Sia [12] proposes a 'Fixed-Length Feature Construction with Substitution' method called FLFCWS. It constructs a set that consist of randomly combined feature subsets. This allows initial features to be used more than once for feature construction.

A. Contribution

We propose an automated algorithm that is able to systematically construct and assess suitable new features out of data sequences for binary classification tasks. It thereby is also able to utilize the time dimension in a sequence of actions in order to access information, which can have a significant impact on the discriminatory power of features. Thereby, the algorithm transforms sequential data into tuple-based data in a way, that allows standard algorithm such as Neuronal Networks, Bayesian Belief Network, Decision Trees or Naive Bayes to be applied on sequential data.

So far, feature construction techniques build new features 'horizontally' by combining columns of a data set. We also apply this techniques with a larger variety of mathematical operators. In addition to that we include the time elapsed between data points. Our approach is novel, since we try to 'vertically' go down the time axis of a sequence and create features by combining numeric values (or its probabilities in terms of string attributes) of the corresponding occurrences. The original values are aggregated during the feature construction process and this allows to store sequence based information on tuple level. As a result of that, the above mentioned standard algorithms can be applied (not all are able to handle sequenced data sets).

III. General characteristics of sequential data

This work often refers to the term sequential data. Thereby we understand data, that can be ordered by time and can be attributed to logical units (i.e., the sequence). A simple example for that are sessions in an online shop. Customers can view products and put them into their shopping basket. Every action can be represented in a data base as a row r with several attributes $a_i \in E$. Each row is provided with a timestamp t. A row can be associated to a logical unit s_{id} (in our case the session id). There are n sequences s_{id_n} in a data set E. Each sequence s_{id_n} consist of at least one row r. The number of rows in a sequence equals to the length of a sequence ls, so that $1 \leq ls \leq m$. Table I, depicts the general schema of sequential data: It is important

TABLE I: Schema of sequence data

r	t	s_{id}	a_1	a_2	...	a_i	s_{label}
r_1	t_1	s_{id_1}	a_{1_1}	a_{2_1}	...	a_{i_1}	0
r_2	t_2	s_{id_1}	a_{1_2}	a_{2_2}	...	a_{i_2}	0
r_3	t_3	s_{id_1}	a_{1_3}	a_{2_3}	...	a_{i_3}	0
r_4	t_4	s_{id_2}	a_{1_4}	a_{2_4}	...	a_{i_4}	1
r_5	t_5	s_{id_2}	a_{1_5}	a_{2_5}	...	a_{i_5}	1
r_6	t_6	s_{id_2}	a_{1_6}	a_{2_6}	...	a_{i_6}	1
...	...	...	...	...	...	...	...
r_m	t_m	s_{id_n}	a_{1_m}	a_{2_m}	...	a_{i_m}	...

to differentiate between the number of rows (or tuples) m of a data set and the number of sequences n. Sequence s_{id_1}, for example, has a length ls of three and contains a matrix such as $s_{id_1} = \begin{Bmatrix} a_{1_1} & a_{2_1} & \dots & a_{i_1} \\ a_{1_2} & a_{2_2} & \dots & a_{i_2} \\ a_{1_3} & a_{2_3} & \dots & a_{i_3} \end{Bmatrix}$

In order to use our proposed method, which is described in detail in the following section, the user has also to mark the following columns on a data set:

- t: timestamp column that is used for temporal based features. It is used to calculate the time elapsed between the collected data points of a sequence.
- s_{id}: sequence identifier column that is used for sequence aggregation. It identifies events/objects that can be logically associated to one entity

- s_{label}: the proposed algorithm requires a binary column as target value. This is needed in order to automatically calculate the information gain of newly constructed features. Every sequence must only have one label, i.e., a customer in an online shop is either a returning customer or not (it can not be both at the same time).

During the feature construction process, we will create a feature table, which includes the s_{id}, s_{label} and the created features $f_p \in S$. Please refer to Table II, for a schema of such a table. The data sequences are aggregated on a tuple-based level. This enable the application of many standard classification algorithms.

TABLE II: Schema of feature table

s_{id}	f_1	f_2	...	f_p	s_{label}
s_{id_1}	$f1_1$	$f2_1$	...	f_{p_1}	0
s_{id_2}	$f1_2$	$f2_2$	...	f_{p_2}	1
...	...	...	...	...	...
s_{id_n}	$f1_n$	$f2_n$	...	f_{p_n}	...

IV. Feature Construction for Data Sequences

Our goal is to extend and search the initial problem space as much as possible. Problem space is thereby defined through the primitive (original) attributes E, which are used to solve a binary classification task. The accessible feature space expands, if more features are constructed. Albeit, this leads to an increase in search time, it brings a higher chance to find discriminatory features. In order to keep things as simple as possible, we describe the algorithm in four different subsections, each describing a certain sort of features creation technique. Please note that the initial attributes are, in a first step, categorized in string and numeric attributes. Reason for this is, that not all described functions are applicable on string values. Please note, that after each feature construction technique, we normalize the newly generated features with min-max normalization, depicted in (1). This provides an easy way to compare values that are on different numerical scales or different units of measure.

$$Normalized(e_i) = \frac{e_i - E_{min}}{E_{max} - E_{min}}, for\ E_{max} > E_{min} \quad (1)$$

The first Subsection IV-A will show construction techniques for both string and numeric attributes. The second Subsection IV-B describes construction techniques for string-only attributes. After that we will focus in the third Subsection IV-C on numeric-only construction techniques. Subsection IV-D concludes this section by describing temporal based feature construction techniques.

A. Distinct occurrences based features

The general idea for this type of technique is to analyze if different occurrences per sequence allows to discriminate between the given labels. Basically, we aggregate all sequences s_{id_n} and count the distinct occurrences (so no duplicates are counted) for each given string as well as numeric attribute a_{i_n}. The constructed features f_p are than collected in S, together with its corresponding sequence identifier s_{id} and the corresponding session label s_{label}. Please note that the sequence identifier s_{id} is unique in S (as opposed to E). The corresponding pseudo-code is depicted in Fig. 1.

Input: E // set of string and numeric attributes
$s_{label} \in \{0, 1\}$ // single value label indication
Def: $a_i \in E$ // single attribute or a column in a data set
$s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows r
$S = \emptyset$ // set of constructed features
for each $a_i \in E$ {
  for each $s_{id} \in E$ {
    $f_p := (|\{a_{i_n}\}|, s_{id}, s_{label})$
    $S := S \cup f_p$
    }
  }
return S

Fig. 1: Pseudo-code feature construction based on distinct occurences per label

In order to assess the quality of the new constructed feature f_i, we calculate the average of all aggregated values per label $s_{label} \in \{0, 1\}$. The difference between both averages is called split and is calculated as depicted in (4).

$$avg_0 = \quad avg(\{f_p \in S | s_{label} = 0\}) \quad (2)$$

$$avg_1 = \quad avg(\{f_p \in S | s_{label} = 1\}) \quad (3)$$

$$split_{f_i} = \quad \frac{|avg_0 - avg_1|}{avg_0 + avg_1} \quad (4)$$

B. Concatenation based features

Purpose of this type of feature construction is to highlight simpler interactions among data. We systematically concatenate every string attribute in pairs of two and then again, count the distinct value-pairs per sequence identifier. Thereby interactions such as, if a_1 AND a_2 have low value-pair variety for label 0, but a high value-pair variety for label 1, are highlighted. Even for data sets with a high number of different occurrences, this kind of feature construction will highlight distinct occurrences between both labels. This procedure is only applicable on string attributes. This approach is similar to most common column combinations that is described widely in the literature (e.g., [4], [7], [11]). However, we once again use this technique on a different abstraction layer since we aggregate via the sequence identifier s_{id}. The corresponding pseudo-code is depicted in Fig. 2.

The algorithm copies the input attribute list E for looping purposes into a second variable E_2. Right after the second loop, it deletes the current attribute from copied list ($E_2 - a_{2i}$). Reason for this is to avoid the same features to occur twice, due to symmetric properties. If, for example, we combine column $a_i = X$ and $a_j = Y$ of a data set, we will yield feature XY. This feature will have the same variability per sequence as the vice versa feature YX. The construction of such features can be avoided by deleting the current feature from the copied feature list E_2.

Input: E // set of primitive string attributes
$s_{label} \in \{0, 1\}$ // single value label indication
Def: $a_i \in E$ // single attribute or a column in a data set
$s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows r
$S = \emptyset$ // set of constructed features
$E_2 = E$ // copy of E, used for looping
$con()$ // concatenates two values
for each $a_i \in E$ {
//remove a_i to avoid vice versa features
$E_2 := E_2 - \{a_i\}$
for each $a_j \in E_2$ {
for each $s_{id} \in E$ {
$f_p = (|(con(a_i, a_j))|, s_{id}, s_{label})$
$S = S \cup f_p$
}
}
}
return S

Fig. 2: Pseudo-code feature construction based on concatenated string attributes

Input: E // set of primitive numeric attributes
$s_{label} \in \{0, 1\}$ // single value label indication
Def: $a_i \in E$ // single attribute or a column in a data set
$s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows r
$S = \emptyset$ // set of constructed features
$E_2 = E$ // copy of E, used for looping
O // set of arithmetic operators
ls // length of a sequence s_{id}
for each $a_i \in E$ {
//remove a_i to avoid vice versa features
$E_2 := E_2 - \{a_i\}$
for each $a_j \in E_2$ {
for each $o \in O$ {
for each $s_{id} \in E$ {
$f_p = (\sum_{i=1}^{ls}(a_i \ o \ a_j), s_{id}, s_{label})$
$S = S \cup f_p$
}
}
}
}
return S

Fig. 3: Pseudo-code feature construction based on numeric attributes

C. Numeric operator based features

The basic idea of this feature construction technique is to combine two numeric attributes with basic arithmetic operators such as "+", "-", "*" or "/". Garcia [9] and Pagallo [6] for instance are using similar techniques with fewer operators. In addition to the repeated use of arithmetic operators we, once again, use the sequence identifier attribute to aggregate the constructed features for each sequence. Lets put this into an example: attributes a_i and a_j are combined with the multiplication operator "*" for a sequence s_{id_1}. The resulting feature $f = a_i * a_j$ exists in the sequence $s_{id_1} = \begin{Bmatrix} a_{i_1} & a_{j_1} \\ a_{i_2} & a_{j_2} \\ a_{i_3} & a_{j_3} \end{Bmatrix}$ The sequence consists of three data points. In the aggregation phase, we sum up the multiplied attributes for all sequences $\sum_{j=1}^{3} f_{ij}$. This process is repeated for all possible combinations of numeric attributes for all of the above mentioned mathematical operators. The full pseudo-code is depicted in Fig. 3. For these technique, we also avoid vice versa features as described in previous Subsection IV-B.

D. Temporal axis based features

The general idea for this feature construction technique is to use the time axis, which is displayed in each sequence by the time indicator column t. This is applicable for both, numeric as well as string attributes. However, for string attributes, there needs to be some preparations done, which are explained further down in this subsection. We continue here to describe the process for numeric attributes. What the algorithm basically does, is to multiply the time interval (e.g., days, hours, minutes), between earliest data point and the current data point with the numeric value of corresponding attribute, which results in a weighting.

Table III, shows this for two example sequences. We have two attributes a_i and a_j for two sequences as well as the t column. In order to calculate the temporal based feature for attribute sequence $s_{id} = 1$ in terms of attribute a_i, we first have to calculate the time between the earliest data point of $s_{id} = 1$ and each of the 'current' data points. In Table III, this is depicted by the $\Delta time\_in\_days$ column. The next step is to multiply the value of each t_i in $s_{id} = 1$ with its corresponding delta time value: $(a_{i_1} * 1, a_{i_2} * 10, \ldots, a_{i_4} * 23)$. The sum of this value is the new time based constructed feature f_p. This process is repeated for all sequences s and for all numerical attributes E.

TABLE III: Example for creating temporal based features

s_{id}	t	$min(t)$	$\Delta time-in\ days$	a_i	a_j	s_{label}
1	01.01.2013	01.01.2013	1	a_{i_1}	a_{j_1}	0
1	10.01.2013	01.01.2013	10	a_{i_2}	a_{j_2}	0
1	15.01.2013	01.01.2013	15	a_{i_3}	a_{j_3}	0
1	23.01.2013	01.01.2013	23	a_{i_4}	a_{j_4}	0
2	24.01.2013	24.01.2013	1	a_{i_5}	a_{j_5}	1
2	28.01.2013	24.01.2013	4	a_{i_6}	a_{j_6}	1
2	30.01.2013	24.01.2013	6	a_{i_7}	a_{j_7}	1

However, there are two directions of including the time for this feature construction technique. What we described above puts a stronger emphasis on the recent history. It is also possible to increase the weight of the past by using the (max_date - current_date) operator to calculate the $\Delta time\_in\_days$ column.

The above mentioned techniques are applicable on numeric

 ISBN: 978-1-61208-334-6

Input: E // set of primitive numeric attributes
t // time indicator column
$s_{label} \in \{0, 1\}$ // label indication
Def: $a_i \in E$ // single attribute or a column in a data set
$s_{id} = (r_1, r_2, \ldots, r_m)$ // sequences of rows r
$S = \emptyset$ // set of constructed features
$E_2 = E$ // copy of E, used for looping
ls // length of a sequence s_{id}
$max()$ // returns max value of a set
for each $a_i \in E$ {
for each s_{id} {
$f_p = (\sum_{i=1}^{ls}((\max_{k=1,\ldots,ls}(t_k) - t_i) * a_i), s_{id}, s_{label})$
$S = \{S \cup f_p\}$
}
}
return S

Fig. 4: Pseudo-code feature construction of temporal based attributes

attributes. For string attributes, it is possible to replace the string by the posterior probability $p(\theta|x)$ (see also Hand [14], pp. 117-118 and pp. 354-356). Thereby θ represents the probability of the parameters for a given evidence x. In our example case, we have the distribution of our two labels as parameters θ and occurrences of a_i as evidence x. Based on this the posterior probability can be calculated as depicted in (5)

$$p(s_{label} = 1|a_i) = \frac{p(a_i|s_{label}=1) * p(s_{label}=1))}{p(a_i)} \quad (5)$$

In order to apply this for string based attributes, we can construct new features f for string attributes as depicted in (6)

$$f_p = \sum_{i=1}^{ls}(\max_{k=1,\ldots,m}(t_k) - t_i) * (p(s_{label} = 1|a_i)) \quad (6)$$

If there are occurrences in the data that have a great tendency towards a particular label (i.e., having a high possibility for one label), we can make this pattern visible by multiplying the posterior possibility with the temporal axis of the given sequence.

However, if there are too many different occurrences, lets say more than 1.000 different values per attribute, this technique could have problems dealing with very small probabilities. So, it is recommended to take the logarithm of the posterior probability for cases with high cardinality.

V. Experimental setup and Results

This section is divided into three subsection in which we will first look at the technical framework we used during our experiments. This is followed by a brief look at the data profile and the corresponding classification task. The third subsection will then compare and discuss the results of our experiments.

A. Technical Framework and Infrastructure

All implementations and experiments were carried out on a Microsoft Windows Server 2008 R2 Enterprise Edition (6.1.7601 Service Pack 1 Build 7601) with four Intel Xeon CPUs E5320 (1.86 GHz, 1862 MHz). The available RAM comprised of 20 GB installed physical memory and 62 GB virtual memory (size of page file 42 GB). The widespread freeware data mining software RapidMiner (version 5.2.008) was used for the standard methods under comparison: Decision Tree, Naive Bayes, Neuronal Network and Random Forrest (for a closer description please also see Witten [13] pp. 191-294, Han [1] pp. 291-337). The method Bayesian Belief Network required the installation of the free RapidMiner extension WEKA. We used the default parameters for all of the above mentioned classification algorithms.

B. Data Profile

The data we used for our experiments was retrieved from the DataMiningCup 2013. The training as well as the test data set can be downloaded on the following site: 'http://www.data-mining-cup.de/en/review/dmc-2013/'. The given historical data from an online shop consisting of session activities from customers. The goal of the task is to classify sessions into a buyer and a non-buyer class. The given training data has the following parameters:

- total number of rows: 429,013
- number of sessions: 50,000
- number of numeric attributes: 21
- number of string attributes: 2

The test data has the following parameters:

- total number of rows: 45,068
- number of sessions: 5,111
- number of numeric attributes: 21
- number of string attributes: 2

Most of the given attributes are numeric. Please note that there is no exact time column given. Therefore, we used a artificial id column to map the temporal order of the various sessions. We also used this column to calculate the temporal based features described in Subsection IV-D.

C. Comparison of original attributes vs constructed features sets

As a first step, we used the given primitive attributes to solve the task. We used the accuracy measurement (7) due to a similar label distribution (45 % to 55 %) and both labels are associated with the same 'costs' for misclassification.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

As it can be seen in Fig. 6, the Naive Bayes classification algorithm was able to achieve better result than the base line (the other algorithms defaulted and predicted label = 0 for all sessions). The Bayesian Belief Networks are not applicable for situations in which the same s_{id} can occur several time (therefore a accuracy rate of 0 %). In a next step, we used our

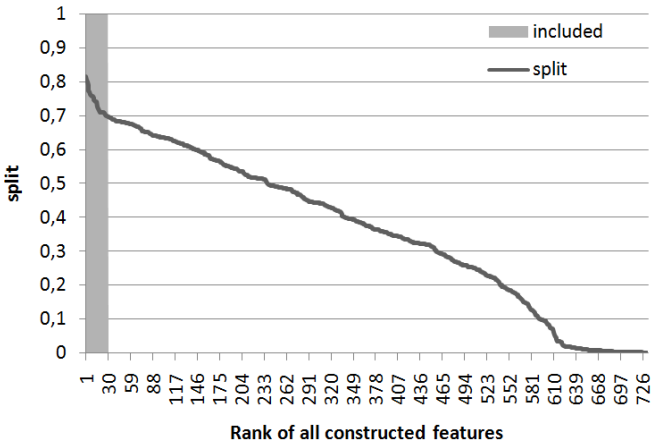

Fig. 5: All constructed features ranked by their split value.

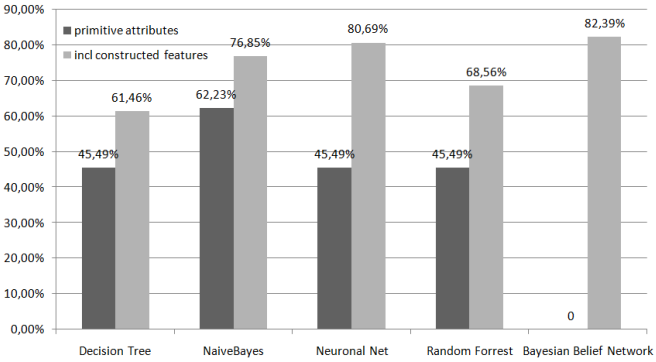

Fig. 6: Accuracy rate comparison original data set with primitive attributes and the same algorithms applied including the top 29 constructed features.

suggested feature construction algorithm in order to aggregate the sessions and find useful features. During this process, a grand total of 732 features were created:

- # of distinct occurrences based features: 23
- # of string concatenation based features: 2
- # of arithmetic based features: 686
- # of temporal axis based features: 21

All features were normalized with the min-max normalization and assessed by calculating the split value for each feature. The features were ranked by their split value, as it can be seen in Fig. 5. The best feature achieve a split value of 0.815, the lowest of 0.0003. In order to keep execution times low, we chose only the top 29 constructed features for our second run. Fig. 6 shows the impressive improvement for the compared standard methods. Since the s_{id} is unique for the constructed features set, the Bayesian Belief Networks are applicable.

VI. Conclusion

Data pre-processing and selection are very important steps in the data mining process. This can be challenging, if there is no domain expert knowledge available. The algorithm proposed in this work helps, not only to understand the patterns within the data, but also, to simplify more complex data structures (such as sequential data). It can be applied in conjunction with well known standard algorithms and can boost classification performance in a big variety of fields with similar specifications (such as the detection of credit card fraud, network intrusions, bots in computer games). Its systematic approach can also help domain experts to find previously unknown interactions among data and therefore, to get a better understanding of their domain.

VII. Future Work

Further ways for extending the features space could be to implement more numerical features generated by logarithm, exponential powers or combinations of more than two attributes. The algorithm itself could be optimized to assess the quality of a candidate feature before actually calculating it. Another development direction could be to align the constructed features in a way, that would allow to classify data without the help of one of the standard algorithms.

References

[1] J. Han and M. Kamber, "Data mining: Concepts and techniques" 2. edition pp. 48-97 second edition, San Francisco, Morgan Kaufmann, 2006

[2] H. Liu and H. Motoda, "Feature Extraction, Construction and Selection: A Data Mining Perspective", Boston, Kluwe Academic Publisher, 1998

[3] L. S. Shafti and E. Pérez "Constructive Induction and Genetic Algorithms for Learning Concepts with Complex Interaction", in Proceedings of The Genetic and Evolutionary Computation Conference, Washington, June 2005, pp. 1811-1818

[4] L. S. Shafti and E. Pérez "Data Reduction by Genetic Algorithms and Non-algebraic Feature Construction: a Case Study", in Proceedings of: Eighth International Conference on Hybrid Intelligent Systems, Barcelona, September 2008, pp. 573-578

[5] R. Setiono and H. Liu "Fragmentation Problem and Automated Feature Construction", in Proceedings of: fourth Conference on Data Mining and Optimization (DMO), Langkawi, September 2012, pp. 53-58

[6] G. Pagallo "Learning DNF by Decision Trees", Machine Learning, pp. 71-99 Kluwer Academic Publishers, 1990

[7] B. Zupan and M. Bohanec "Feature Transformation by Function Decomposition", in Journal IEEE Intelligent Systems archive Volume 13 Issue 2, March 1998, pp. 38-43

[8] J.R. Quinlan "C4.5: Programs for Machine Learning". Morgan Kaufmann, 1993

[9] D. García, A. González and R. Pérez, "A Two-Step Approach of Feature Construction for a Genetic Learning Algorithm", in Proceedings of: IEEE International Conference on Fuzzy Systems, Taipei, June 2011, pp. 1255-1262

[10] D. García, Antonio González and R. Pérez, "An Iterative Strategy for Feature Construction on a Fuzzy Rule-Based Learning Algorithm", in Proceedings of: 11th International Conference on Intelligent Systems Design and Applications, Cordoba, November 2011, pp. 1235-1240

[11] R. Alfred "DARA: Data Summarisation with Feature Construction", in Proceedings of: Second Asia International Conference on Modelling & Simulation, Kuala Lumpur, May 2008, pp. 830-835

[12] F. Sia and R. Alfred "Evolutionary-Based Feature Construction with Substitution for Data Summarization using DARA", in Proceedings of: fourth Conference on Data Mining and Optimization (DMO), Langkawi, September 2012, pp. 53-58

[13] I. Witten and F. Eibe, "Data mining : practical machine learning tools and techniques" 2. edition, San Francisco, Morgan Kaufmann, 2005, pp. 48-97

[14] D. Hand, H. Mannila and P. Smyth "Principles of Data Mining", MIT Press, 2001

 ISBN: 978-1-61208-334-6